Discrete Applied Mathematics 42 (1993) 147–175
North-Holland

Efficient reformulation for 0-1 programs – methods and computational results

B.L. Dietrich, L.F. Escudero

IBM Research, T.J. Watson Research Center, Yorktown Heights, NY, USA

F. Chance

Cornell University, Ithaca, NY, USA

Received 15 June 1990
Revised 1 March 1991

Abstract

Dietrich, B.L., L.F. Escudero and F. Chance, Efficient reformulation for 0-1 programs – methods and computational results, Discrete Applied Mathematics 42 (1993) 147–175.

We introduce two general methods for 0-1 program reformulation. Our first method generalizes coefficient reduction, our second method generalizes lifting. Together they provide a unifying interpretation of many previously described automatic reformulation methods. The particular model structures that we consider are individual knapsack constraints, pairs of knapsack constraints, clique and cover induced inequalities, variable upper bounding constraints and capacity expansion constraints. We describe several easy applications of our reformulation procedures. Some computational experience is reported, including the currently best known results on a well-known 147×2655 benchmark problem.

Keywords: 0-1 programs, knapsack constraints, cutting planes, capacity expansion, variable upper bounding constraints, maximal cliques, minimal covers, coefficient reduction and increase.

Correspondence to: Dr. L.F. Escudero, IBM Research, T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA.

1. Introduction

Consider the 0-1 program

$$\begin{aligned} z &= \text{MAX } gx, \\ \text{s.t.} \quad & Ax \leq b, \\ & x_j \in \{0,1\},\ j \in J, \end{aligned} \tag{1.1}$$

where x is the column vector of the set J of 0-1 variables, g is the related row vector of the objective function, A is the coefficient matrix of the constraints, and b is the right-hand side (henceforth rhs). All vectors are assumed to have the appropriate dimensions. (The LP relaxation of (1.1) is the same system (1.1) where each x_j is allowed to take any value in the range of $[0, 1]$.) Let I denote the set of constraints, and let $J_i \subseteq J$ be the set of variables with nonzero coefficient in constraint $i \in I$.

We consider knapsack constraints of the form

$$\sum_{j \in J_r} a_{r,j} x_j \leq b_r, \tag{1.2}$$

where $r \in I$ and the coefficients $0 < a_{r,j} \leq b_r$. When no ambiguity will arise we will drop the constraint index r. Any constraint on 0-1 variables can be put in form (1.2) by replacing x_j by its binary complement $1 - x_j$ and appropriately modifying the rhs b whenever $a_j < 0$. Once all coefficients are nonnegative in a given knapsack, if $a_j > b$, then the variable x_j can be fixed at zero, while if $\sum_j a_j \leq b$, the constraint is trivially satisfied by any 0-1 solution, and can be dropped. (We assume that problem (1.1) has been preprocessed and properly reduced, see [20].) Obviously it need not be possible to simultaneously transform all the constraints in (1.1) into knapsack form, e.g., if there exists $j \in J$ and $r, s \in I$ such that $a_{r,j} a_{s,j} < 0$. This observation is especially important in the context of this paper, since our methods replace knapsack constraints by using additional information provided by other constraints in the problem.

Two sets of inequalities $Ax \leq b$ and $A'x \leq b'$ are said to be 0-1 equivalent if they admit exactly the same set of 0-1 solutions. In the context of 0-1 problems, we say that the set of constraints $A'x \leq b'$ is *as tight as* the set $Ax \leq b$ whenever

$$\{x \in [0,1]^n \mid A'x \leq b'\} \subseteq \{x \in [0,1]^n \mid Ax \leq b\}, \tag{1.3}$$

where $n \equiv |J|$. We say that $A'x \leq b'$ is *tighter* than $Ax \leq b$ if the containment (1.3) is proper. Our goal is finding tight formulations for (1.1) without expending excessive computational effort or adding many additional constraints. Facet-defining inequalities have been studied in [1, 23, 34, 37, 44, 45] among others. See in [35, 36] good surveys on the topic.

Tight integer equivalent inequalities can be obtained by coefficient reduction and lifting techniques that exploit only the special structure of individual 0-1 knapsack constraints, see [18, 27–29, 33] among others. In this paper we propose a methodology

for coefficient reduction on knapsack constraints that uses additional information derived from other constraints on the 0-1 variables. Our procedures can be used within the framework described in [8, 18, 24, 38] for an LP cutting plane based solution procedure for 0-1 programs. We will replace individual constraints such as (1.2), and append (or maintain as a separate list) additional implied inequalities as needed.

A clique is a subset $C \subseteq J$ such that at most one of the associated variables can have a nonzero value in any feasible solution. That is, the inequality (1.4) is satisfied by any feasible solution of (1.1).

$$\sum_{j \in C} x_j \leq 1. \tag{1.4}$$

Also the trivial constraint $x_l \leq 1$ can be considered a clique constraint induced by $C = \{l\}$. In addition to tightening the LP relaxation, identifying nontrivial cliques has proved useful in branch-and-bound methodologies (see [4, 8, 18, 22, 40, 43]).

A cover is a subset $C \subseteq J$ such that at least $|C| - k_C$ of the associated variables must take the value zero in any feasible solution. Associated with each cover is the valid inequality

$$\sum_{j \in C} x_j \leq k_C, \tag{1.5}$$

where $k_C < |C|$. Note that cliques are covers with $k_C = 1$.

A cover inequality (1.5) is said to be implied by the knapsack constraint (1.2) if for each $C' \subseteq C$ having $|C'| > k_C$, the sum of coefficients $\sum_{j \in C'} a_{r,j}$ exceeds the rhs b_r. Under some circumstances a cover inequality implied by a knapsack constraint can be a facet of the knapsack polytope *convex hull* $\{x \mid \sum_{j \in J_r} a_{r,j} x_j \leq b_r, x \in \{0,1\}^n\}$, where $n \equiv |J_r|$. See [35] for a good survey on this topic.

A *minimal cover* is a cover C such that if any index $j \in C$ is deleted from the inequality (1.5), the resulting inequality is redundant. For a minimal cover, the induced inequality (1.5) must have $k_C = |C| - 1$. Several methods have been proposed (see [8, 41]) for identifying (lifted) minimal cover induced inequalities that are most violated by the current LP solution. These constraints are then appended to the original problem. See also [15, 22]. Interestingly enough clique and cover induced cuts and the inequalities obtained by reducing individual coefficients (see [27]) can be derived as Gomory–Chvatal cuts [7, 21] by using appropriate multiplier vectors; see [12].

Apart from identifying and appending violated clique and cover induced inequalities at each optimal LP iteration as in [8, 41], we have found it useful to identify all maximal cliques and a subset of (lifted) minimal covers with reasonably small k_C (see [15]), and append the induced inequalities to the original problem. In addition to immediately tightening the LP relaxation of the original 0-1 program and reducing the size of the corresponding branch-and-bound tree, these cliques and covers can be used in our other reformulation methods to further tighten the LP (see below). Our techniques are specially promising in the new branch-and-cut methodology; see, e.g. [38].

The paper is organized as follows. Section 2 presents our "ideal" reformulation of 0-1 programs. It can be viewed as a unifying interpretation of the reformulation techniques in our previous reports [9–16]. Section 3 presents efficient methods for reducing individual coefficients in knapsack constraints by sequentially exploiting the information derived from other knapsack constraints, using additional structures such as trivial cliques (Section 3.1), variable upper bounds (Section 3.2) and capacity expansion constraints (Section 3.3). Section 4 demonstrates simultaneous reduction of coefficients associated with cliques and covers. Section 5 presents methodology for increasing coefficients associated with knapsacks and covers. The effectiveness of the procedures is illustrated by applying them to examples, most of which are taken from the literature. Section 6 reports promising computational experience, including impressive results on a well-known real-life problem. Finally, we offer some conclusions.

2. "Ideal" reformulation

We present two forms of "ideal" reformulation, each of which involves modifying individual constraints by considering implications of the remaining constraints. Our first method, a generalization of "coefficient reduction", tightens a constraint by reducing both the rhs and selected coefficients. (See [9, 10, 18, 27–29, 33] for various coefficients reduction results.) Our second method tightens a constraint, by increasing the coefficient of selected variables, while keeping the rhs fixed. It is most easily applied to cover inequalities.

Consider the knapsack constraint (1.2) and assume that a cover inequality (1.5) with $C \subseteq J_r$ has been identified. Let $R_{r,C}$ denote the maximum value that can be taken by the left-hand side (henceforth lhs) of constraint r in a feasible solution to (1.1) if cover inequality (1.5) is strictly satisfied. This value is given by

$$R_{r,C} = \text{MAX} \sum_{j \in J_r} a_{r,j} x_j, \tag{2.1}$$

$$\text{s.t.} \quad \sum_{j \in J_i} a_{i,j} x_j \leq b_i, \ \forall i \in I - r, \tag{2.2}$$

$$\sum_{j \in J_r} a_{r,j} x_j \leq b_r, \tag{2.3}$$

$$\sum_{j \in C} x_j \leq k_C - 1, \tag{2.4}$$

$$x_j \in \{0, 1\}, \ \forall j \in J. \tag{2.5}$$

If $R_{r,C} < b_r$ then we can modify (1.2) to obtain an equivalent but tighter formulation of (1.1). We modify (1.2) by reducing the coefficients $a_{r,j}$ with $j \in C$ and the rhs b_r. For any choice of $\Delta^-_{r,C}$ such that

$$0 \leq \Delta^-_{r,C} \leq b_r - R_{r,C} \tag{2.6}$$

we modify (1.2) as follows.

$$\bar{b}_r = b_r - \Delta^-_{r,C},$$

$$\bar{a}_{r,j} = \begin{cases} a_{r,j} - \dfrac{\Delta^-_{r,C}}{k_C}, & j \in C, \\ a_{r,j}, & j \in J_r - C. \end{cases} \tag{2.7}$$

The reduced coefficient constraint is given by

$$\sum_{j \in J_r} \bar{a}_{r,j} x_j \leq \bar{b}_r. \tag{2.8}$$

Theorem 2.1. *Replacing the rth knapsack constraint* (1.2) *by* (2.8) *in* (1.1), *where the coefficients* $\bar{a}_{r,j}$ *and* $\bar{b}_r$, *are given by* (2.6) *and* (2.7), *and appending the cover inequality* (1.5) *whenever it is implied by* (1.2), *results in a* 0-1 *equivalent formulation. The new formulation is tighter than the original whenever* $\Delta^-_{r,C} > 0$.

Proof. First note that the LP formed by the set of constraints (1.5), (2.2) and (2.8), together with the upper and lower bounds $0 \leq x_j \leq 1$, $\forall j \in J$, is as tight as the LP relaxation of the original 0-1 program. To see this we show first that any infeasible solution x for the LP relaxation of (1.1) also violates one of the constraints (1.5), (2.2) and (2.8). If x violates one of the constraints (1.5) or (2.2) in the LP relaxation of (1.1) the result is obvious so assume that x violates (2.3). If $\sum_j \bar{a}_{r,j} > \bar{b}_r$ then (2.8) is violated. Otherwise, $(\Delta^-_{r,C}/k_C) \sum_{j \in C} x_j = \sum_{j \in J_r} (a_{r,j} - \bar{a}_{r,j}) x_j > b_r - \bar{b}_r = \Delta^-_{r,C}$. But then $\sum_{j \in C} x_j > k_C$, violating (1.5). In fact, for $\Delta^-_{r,C} > 0$, the set of constraints (1.5), (2.2) and (2.8), together with the upper and lower bounds on the variables, is tighter than the LP relaxation of (1.1). The cover inequality eliminates solutions x having $\sum_{j \in C} x_j > k_C$, while constraint (2.8) eliminates those solutions satisfying the following two conditions

$$\sum_{j \in C} x_j < k_C, \tag{2.9}$$

$$b_r - \sum_{j \in J_r} a_{r,j} x_j < \frac{\Delta^-_{r,C}}{k_C} \left(k_C - \sum_{j \in C} x_j \right). \tag{2.10}$$

To see that the new set of constraints is 0-1 equivalent to the constraint set in (1.1), let x be a feasible solution to the original 0-1 program. Then, since x satisfies the inequality (1.5), we need only consider the two alternatives $\sum_{j \in C} x_j \leq k_C - 1$ and $\sum_{j \in C} x_j = k_C$. In the first case, by the definition of $\bar{a}_{r,j}$, $\bar{b}_r$, and $R_{r,C}$ we easily verify that x satisfies (2.8); note that $\sum_{j \in J_r} \bar{a}_{r,j} x_j \leq \sum_{j \in J_r} a_{r,j} x_j \leq R_{r,C} \leq \bar{b}_r$. For the second case we have $\sum_{j \in J_r} \bar{a}_{r,j} x_j = \sum_{j \in J_r} a_{r,j} x_j - (\Delta^-_{r,C}/k_C) \sum_{j \in C} x_j = \sum_{j \in J_r} a_{r,j} x_j - \Delta^-_{r,C} \leq b_r - \Delta^-_{r,C} = \bar{b}_r$. On the other hand, suppose that the solution x satisfies (1.5) and (2.8). Then substituting x into (1.2) we have

$$\begin{aligned} \sum_{j \in J_r} a_{r,j} x_j &= \sum_{j \in J_r} \bar{a}_{r,j} x_j + \frac{\Delta^-_{r,C}}{k_C} \sum_{j \in C} x_{r,j} \leq \bar{b}_r + \frac{\Delta^-_{r,C}}{k_C} \sum_{j \in C} x_{r,j} \\ &\leq \bar{b}_r + \Delta^-_{r,C} = b_r. \end{aligned} \tag{2.11}$$

The last inequality follows from (1.5), the remaining equalities and inequalities follow from the definitions in (2.7) and (2.8). □

We often choose to restrict $\Delta^-_{r,C}$ so that the coefficients remain nonnegative. That is $\Delta^-_{r,C} = \text{MIN}\{b_r - R_{r,C}, \text{MIN}\{k_C a_{r,j},\ j \in C\}\}$ (then, $C \subseteq J_r$). Otherwise, if a coefficient becomes negative and the coefficient reduction process on that constraint is to be continued, the corresponding variable must be replaced by its binary complement.

If the cover inequality (1.5) was implied by constraints in $I - r$ then 0-1 equivalence holds without appending it. If it was implied only by constraint r, and the reduction is large, the resulting constraint (2.8) may no longer imply (1.5), but it is usually possible to select $\Delta > 0$, so that (1.5) remains implied, and the formulation is tightened; see [13]. However, we recommend maintaining a list of all detected cover inequalities, and appending them as needed throughout the solution process.

The second method for constraint reformulation increases the coefficients in a knapsack constraint (1.2) corresponding to variables in some (possibly, trivial) cover inequality (1.5) as follows. For each $l \in C$ solve the following 0-1 program.

$$S^l_{r,C} = \text{MAX} \sum_{j \in J_r} a_{r,j} x_j, \tag{2.12}$$

$$\text{s.t.} \quad \sum_{j \in J_i} a_{i,j} x_j \leq b_i, \ \forall i \in I - r, \tag{2.13}$$

$$\sum_{j \in J_r} a_{r,j} x_j \leq b_r, \tag{2.14}$$

$$x_l = 1, \tag{2.15}$$

$$x_j \in \{0,1\}, \ \forall j \in J. \tag{2.16}$$

Then $S^l_{r,C}$ is the maximum value that the lhs of (1.2) can take, if x is feasible and the variable x_l is set to one. (Note that it is not required that $l \in J_r$.) If $S^l_{r,C} < b_r$, then we can increase $a_{r,l}$ as follows.

$$\Delta^{+l}_{r,C} = (b_r - S^l_{r,C}) \div k_C, \tag{2.17}$$

$$\bar{a}_{r,j} = \begin{cases} a_{r,j} + \Delta^{+j}_{r,C}, & j \in C, \\ a_{r,j}, & j \in J_r - C. \end{cases} \tag{2.18}$$

This produces the constraint

$$\sum_{j \in J_r \cup C} \bar{a}_{r,j} x_j \leq b_r. \tag{2.19}$$

Note that $\bar{a}_{r,j}$ may become positive for $j \in C - J_r$. Because we divide the slack by k_C, constraint (2.19) can be weak and sometimes it is better to use several (possibly, trivial) cliques rather than one cover.

Theorem 2.2. *Replacing the rth knapsack constraint* (1.2) *by* (2.19) *in* (1.1), *where*

the coefficients $\bar{a}_{r,j}$ *are given by* (2.18), *and appending* (1.5) *whenever it is implied by* (1.2), *results in a* 0–1 *equivalent formulation. The new formulation is tighter than the original whenever there exists* $l \in C$ *such that* $\Delta^{+l}_{r,C} > 0$.

Proof. It is obvious that, any solution (integer or fractional) that satisfies the new set of constraints also satisfies the original set, and that the new set is tighter if a coefficient is increased. We need only establish that (2.19) does not cut off any feasible solution. Let x be a feasible solution to the original 0-1 program. We show that it also satisfies (2.19), by considering the possible values of $\sum_{l \in C} x_l \leq k_C$. Let $C' \subset C$ be the index set of the variables in C with $x_l = 1$. It is obvious that (2.19) is satisfied when $C' = \emptyset$, so assume $|C'| \geq 1$. Let $J'_r \subset J_r \cup C$ denote the index set of variables in $J_r \cup C$ with $x_j = 1$. The lhs of (2.19) is given by

$$\begin{aligned}\sum_{j \in J'_r} \bar{a}_{r,j} x_j &= \sum_{j \in J'_r} a_{r,j} x_j + \sum_{l \in C'} \Delta^{+l}_{r,C} \\ &= 1/|C'| \sum_{l \in C'} \left[\sum_{j \in J'_r} a_{r,j} x_j + |C'| \Delta^{+l}_{r,C} \right] \\ &\leq 1/|C'| \sum_{l \in C'} [S^l_{r,C} + k_C \Delta^{+l}_{r,C}] = b_r, \end{aligned} \tag{2.20}$$

so x satisfies (2.19). □

The proof also holds for any choice of $\Delta^l_{r,C} \leq \Delta^{+l}_{r,C}$.

The necessary and sufficient condition for applying the first procedure, (2.1)–(2.8), is that $R_{r,C} < b_r$ (i.e., $\Delta^-_{r,C} > 0$). Determining $R_{r,C}$ is generally quite difficult. It involves solving a 0-1 program with perhaps one more constraint than the original problem. Therefore, this method of generating cutting planes is not likely to lead to an efficient algorithm for solving, or even for preprocessing, 0-1 programs. However, upper bounds on $R_{r,C}$ may be quite easy to compute (e.g., LP relaxation). (See [39] for a similar approach in a different context.) Tighter upper bounds can be obtained when the constraint $cx \leq \bar{z}$ is appended to the LP relaxation of (2.2)–(2.5), where $\bar{z}$ is an upper bound on the optimal value z (1.1). ($\bar{z}$ can be the current incumbent solution.) Also when solving (2.1)–(2.5) for different choices of C and r it is not necessary to begin each problem from scratch. The optimal solution of a previous problem can be used as a starting solution for the next problem. Let $\bar{R}_{r,C}$ denote an upper bound on $R_{r,C}$ (2.1)–(2.5). Note that a tighter equivalent constraint can be generated whenever $\bar{R}_{r,C} < b_r$; the new constraint does not cut off the entire region eliminated by the constraint generated using $R_{r,C}$. Therefore, the key step in reformulating 0-1 programs via coefficient reduction is developing good, easy to compute upper bounds for $R_{r,C}$.

As for the first procedure, determining the value of $S^l_{r,C}$ is likely to be quite difficult. It is essentially equivalent to solving the original 0-1 program with one less variable. Any upper bound, say $\bar{S}^l_{r,C}$, on $S^l_{r,C}$ can be used in place of $S^l_{r,C}$, provided that $\bar{S}^l_{r,C} < b_r$. The LP upper bound is easy to compute, and for certain choices of

C and r, other upper bounds can be computed easily by relaxing a subset of the remaining constraints. We have found it useful to apply this coefficient increase procedure to cover inequalities, whether explicitly included in (1.1) or implied by its constraints. Examples of this type are given in Section 5. We remark that this reformulation methodology generalizes the coefficient increase procedure described in [28, 29] and the lifting methods described in [8, 34, 37, 46] among others. (The constraint (1.2) is said to be *lifted* whenever $a_{r,j}=0$ is replaced by $\bar{a}_{r,j}>0$.)

In the following sections we show how good bounds on $R_{r,C}$ and $S^l_{r,C}$ can be easily computed by exploiting special classes of constraints common in 0-1 programs.

3. Reducing individual coefficients

In all the following we drop the sign $-$ from $\Delta^-_{r,C}$. When no confusion will arise we also drop r and C, and simply denote the reduction by Δ.

3.1. Using trivial cliques

Singleton sets $C=\{l\}$ (i.e., trivial constraints $x_l \le 1$) are considered in [27] to reduce the coefficient $a_{r,l}$ and the rhs b_r using the upper bound $R_{r,\{l\},0}$, where

$$R_{r,\{l\},0}=\sum_{j\in J-l} a_{r,j}. \tag{3.1}$$

That is, all constraints are ignored except the upper bounds on the variables. Computational experience shows that the procedure is effective, particularly in reducing "big M" type coefficients.

One can strengthen this reduction procedure (see [9, 10, 14]) by independently considering other constraints that limit the size of $R_{r,\{l\}}$; see the details in [14]. For any $i\in I$ we let $R_{r,\{l\},i}$ be the optimal value of the following 0-1 knapsack problem (KP).

$$\begin{aligned} R_{r,\{l\},i}&=\text{MAX}\sum_{j\in J_r-l} a_{r,j}x_j,\\ \text{s.t.}\quad &\sum_{j\in J_i-l} a_{i,j}x_j \le b_i,\\ &x_j\in\{0,1\},\ j\in J_r\cup J_i-l. \end{aligned} \tag{3.2}$$

Problem (3.2) is NP-complete, but for sparse problems the set of variables involved, $(J_r\cap J_i)-l$, is likely to be much smaller than J and the algorithms in [19, 31] can be used. See also [32]. The LP relaxation of (3.2) is trivial to solve and can be used as a valid upper bound. See also in [32] other easily computed upper bounds. We use the bound, $\bar{R}_{r,\{l\}}$, on $R_{r,\{l\}}$ given by

$$\bar{R}_{r,\{l\}}=\text{MIN}\{R_{r,\{l\},i}\mid i\in I_{r,\{l\}}\} \tag{3.3}$$

where $I_{r,\{l\}}$ is the subset of constraints in (1.1) that have been considered. We want

to obtain as small an $\bar{R}_{r,\{l\}}$ as possible, but without excessive computational effort, so we consider only constraints $i \in I$ that are likely to provide good bounds. We choose

$$I_{r,\{l\}} = \{r\} \cup \left\{ i \,\middle|\, \sum_{j \in (J_i \cap J_r) - l} a_{i,j} + \sum_{j \in J_i - J_r \mid a_{i,j} < 0} a_{i,j} \geq \eta b_i \right\}, \tag{3.4}$$

where η is a specified tolerance.

The problem (3.2) for $i = r$ is termed subset-sum problem (SSP). It is difficult to solve by using general KP algorithms, but efficient algorithms are available that take into account the special nature of the objective function; see [30, 32]. (Let us drop r from $R_{r,\{l\},r}$.) Note that the SSPs for two variables $p, q \in J$ are very similar, and the solution to one frequently yields an optimal solution to the other. In fact two SSPs differ only in one element; x_p is fixed to zero for the first one and x_q is fixed to zero for the second one. It is often possible to immediately derive the solution for one problem from the solution of another problem, particularly if the difference $a_{r,p} - a_{r,q}$ is small. Specifically, note that $R_{\{q\}} = R_{\{p\}} + \delta$, where $\delta = a_{r,p} - a_{r,q}$, whenever $x_q = 1$ in the optimal solution for $R_{\{p\}}$ and $0 \leq \delta \leq b_r - R_{\{p\}}$. One use of such SSPs would be to select a constraint $r \in I$ which is satisfied at equality by the current LP solution and solve the SSP for each $l \in J_r$ with x_l fractional; frequently, an l with $\Delta_{\{l\}} > 0$ is found. The resulting coefficient reduction constraint will then cut off the current LP solution, see (2.9) and (2.10).

Example 3.1. Consider the following example with 0-1 variables

$$4x_1 + 2x_2 + 4x_3 + 3x_4 + 10x_5 + 3x_6 \leq 19 \quad (r), \tag{3.5}$$

$$3x_1 + 3x_2 + 2x_3 + 3x_4 + 7x_5 + 10x_6 \leq 20 \quad (i). \tag{3.6}$$

For (3.5) and (3.6) we compute $R_{r,\{5\},r} = 16$, $R_{r,\{5\},i} = 14$ and we obtain the inequality

$$4x_1 + 2x_2 + 4x_3 + 3x_4 + 5x_5 + 3x_6 \leq 14 \quad (r). \tag{3.7}$$

For (3.6) and (3.7) we compute $R_{r,\{1\},r} = 14$, $R_{r,\{1\},i} = 12$ and we obtain the inequality

$$2x_1 + 2x_2 + 4x_3 + 3x_4 + 5x_5 + 3x_6 \leq 12 \quad (r). \tag{3.8}$$

Observe that the solution (3.9) is allowed by (3.5) and (3.6) but is infeasible for (3.8).

$$x_1 = x_2 = x_3 = x_4 = 1, \qquad x_5 = 0.4. \tag{3.9}$$

3.2. Using variable upper bounding constraints

In this section we describe procedures for computing tight bounds on $R_{r,\{l\}}$, that consider the implications of variable upper bounding (henceforth VUB) constraints. We assume that J is partioned into a set S of *selection* variables and a set F of *bounding* variables. The selection variables are further partitioned into (not necessarily disjoint) sets S_f for $f \in F$. The variables having x_j for $j \in S_f$ are all bounded by the variable x_f. We first consider VUBs of the form

$$\sum_{j \in S_f} x_j \leq |S_f| x_f \quad f \in F \text{ (or, in disaggregate form,}$$
$$x_j \leq x_f, \ \forall j \in S_f, \ f \in F). \tag{3.10}$$

The polytope defined by the 0-1 system (1.2) and (3.10) is studied in [5]. Such constraints arise in many scheduling and production planning problems; see [17] for an example.

Constraint (3.10) forces to zero all variables x_j, $j \in S_f$ whenever the related bounding variable x_f is made zero. We use this observation to compute an upper bound on $R_{r,\{l\}}$, for $l \in F$. This value, $\bar{R}_{r,\{l\}}$, is given by considering only the knapsack constraint itself and the VUBs, as in the following 0-1 program, which we call SSP with VUBs.

$$\begin{aligned} \bar{R}_{r,\{l\}} = \text{MAX} & \sum_{f \in F-l} \left[a_{r,f} x_f + \sum_{j \in S_f} a_{r,j} x_j \right], \\ \text{s.t.} \quad & \sum_{f \in F-l} \left[a_{r,f} x_f + \sum_{j \in S_f} a_{r,j} x_j \right] \leq b_r, \\ & x_j \leq x_f, \ \forall j \in S_f, \ f \in F-l, \\ & x_j \in \{0,1\}, \ x_f \in \{0,1\}, \ \forall j \in S_f, \ f \in F-l. \end{aligned} \tag{3.11}$$

If $\bar{R}_{r,\{l\}} < b_r$, then both b_r and $a_{r,l}$ can be reduced by the slack $b_r - \bar{R}_{r,\{l\}}$.

Example 3.2. Consider the example (3.12) together with the VUBs (3.13).

$$(3x_1 + 8x_{11} + 9x_{21}) + (3x_2 + 2x_{12} + 3x_{22}) \leq 19, \tag{3.12}$$

$$\begin{aligned} x_{11} \leq x_1, \quad & x_{21} \leq x_1, \\ x_{12} \leq x_2, \quad & x_{22} \leq x_2. \end{aligned} \tag{3.13}$$

We first reduce the rhs and coefficient of x_1 by the slack obtained by using (3.11). Then with the revised inequality, we compute the slack corresponding to x_2, and reduce its coefficient and the rhs. We obtain the constraint

$$(-8x_1 + 8x_{11} + 9x_{21}) + (-4x_2 + 2x_{12} + 3x_{22}) \leq 1. \tag{3.14}$$

The following solution is feasible for (3.12)–(3.13) but violates (3.14).

$$\bar{x}_1 = \bar{x}_{11} = \bar{x}_{21} = 0.5, \qquad \bar{x}_2 = \bar{x}_{12} = \bar{x}_{22} = 0.75. \tag{3.15}$$

A potential computational disadvantage to this method is the dimensions of problem (3.11). A practical algorithm for solving an SSP with VUBs is currently an important open problem. However any upper bound on $\bar{R}_{r,\{l\}}$ could be used. One such bound is given by relaxing the VUBs (3.10) and solving the resulting SSP, but this increases the likelihood that the optimal subset satisfies the capacity constraint at equality — in our case resulting in $\Delta_{r,\{l\}} = 0$. When applying the procedure above to more than one choice of l the total amount of reduction possible depends on the order in which the indices in F are considered.

We can avoid some of the computational difficulties described above by computing a weaker upper bound as follows; see the details in [9]. For each $f \in F$ let $Q_{r,f}$ be the maximum value that the lhs of (1.2) can take if only the variable x_f and those x_j having $j \in S_f$ are allowed to be nonzero. Then $Q_{r,f}$ is given by the following simple SSP with J_f variables.

$$\begin{aligned} Q_{r,f} &= \text{MAX } a_{r,f} + \sum_{j \in S_f} a_{r,j} x_j, \\ \text{s.t.} \quad & \sum_{j \in S_f} a_{r,j} x_j \leq b_r - a_{r,f}, \\ & x_j \in \{0,1\}, \ \forall j \in S_f. \end{aligned} \tag{3.16}$$

Of course, if $a_{r,f} + \sum_{j \in S_f} a_{r,j} \leq b_r$ then (3.16) is trivially solved by setting all the variables involved to one. The coefficient reduction $\Delta_{r,\{l\}}$ is given by

$$\Delta_{r,\{l\}} = b_r - \sum_{f \in F - \{l\}} Q_{r,f} \tag{3.17}$$

and the new coefficients are

$$\begin{aligned} \bar{b}_r &= b_r - \Delta_{r,\{l\}}, \\ \bar{a}_{r,f} &= \begin{cases} a_{r,l} - \Delta_{r,\{l\}}, & f = l, \\ a_{r,f}, & f \in F - \{l\}. \end{cases} \end{aligned} \tag{3.18}$$

Note that for $f \neq l$, since $\bar{b}_r \geq Q_{r,f}$, the optimal solution to (3.16) does not change when its rhs is replaced by $\bar{b} - a_{r,f}$; similarly, with this modification in (3.16), the new value of $Q_{r,l}$ is $Q_{r,l} - \Delta_{r,\{l\}}$. Thus the value of $\Delta_{r,\{l\}}$ does not depend on whether the reduction has already been applied to other coefficients, and so it can be applied to each index in F such that $\Delta_{r,\{f\}} > 0$. In fact, the reduction can be done simultaneously. That is, for each $f \in F$, compute $\Delta_{r,\{f\}}$ (3.17), then let

$$\bar{F} = \{ f \in F \mid \Delta_{r,\{f\}} > 0 \} \tag{3.19}$$

and replace (3.18) by

$$\begin{aligned} \bar{b}_r &= b_r - \sum_{f \in \bar{F}} \Delta_{r,\{f\}}, \\ \bar{a}_{r,f} &= \begin{cases} a_{r,f} - \Delta_{r,f}, & f = \bar{F}, \\ a_{r,f}, & f \in F - \bar{F}. \end{cases} \end{aligned} \tag{3.20}$$

Example 3.3. We apply the above procedure to the example (3.12)–(3.13) by first computing $Q_1 = 12$ and $Q_2 = 8$. Since $\Delta_1 = 3$ and $\Delta_2 = 3$ we obtain the constraint

$$(8x_{11} + 9x_{21}) + (2x_{12} + 3x_{22}) \leq 13. \tag{3.21}$$

Observe that the solution (3.22) is allowed by (3.12) and (3.13), but is infeasible for (3.21).

$$\bar{x}_1 = \bar{x}_{11} = \bar{x}_{21} = 0.75, \qquad \bar{x}_2 = \bar{x}_{12} = \bar{x}_{22} = 0.5. \tag{3.22}$$

Note that the solution (3.15) is *not* eliminated by (3.21). However, the constraint (3.21) implies that either $x_{11}=0$ or $x_{21}=0$, both cannot have the value 1. Our procedure [15] detects that the valid clique inequality (3.23) is violated by the solution (3.22), and appends (3.23) to the LP relaxation of our problem.

$$x_{11}+x_{21}\leq 1. \tag{3.23}$$

3.3. *Using capacity expansion constraints*

As an extension of simple VUBs we consider more general bounding constraints of the form

$$\sum_{j\in S_f} s_{f,j}x_j \leq n_{1,f}+n_{2,f}x_f, \quad f\in F \tag{3.24}$$

where $c_{f,j}$, $n_{1,f}$ and $n_{2,f}$ are all integer, $c_{f,j}>0$, $n_{1,f}<\sum_{j\in S_f} c_j$ and $n_{2,f}>0$. Constraints of this form can arise in many optimization models, particularly resource allocation problems and problems with capacity expansion constraints (henceforth CECs). For these problems, typically a_f is large for $f\in F$ and a_j is small (but positive) for $j\in S_f$.

CEC (3.24) is essentially a knapsack constraint with the rhs determined by the value of x_f. We use this observation to compute an upper bound on $R_{r,\{l\}}$ for $l\in F$. This value, $\bar{R}_{r,\{l\}}$, is given by (3.25), where only the knapsack constraint itself and the CEC are considered.

$$\begin{aligned}
&\bar{R}_{r,\{l\}}=\text{MAX}\sum_{f\in F-l}\left[a_{r,f}x_l+\sum_{j\in S_f}a_{r,j}x_j\right]+\sum_{j\in S_l}a_{r,j}x_j,\\
&\text{s.t.}\quad \sum_{f\in F-l}\left[a_{r,f}x_l+\sum_{j\in S_f}a_{r,j}x_j\right]+\sum_{j\in S_l}a_{r,j}x_j\leq b_r,\\
&\qquad \sum_{j\in S_f}c_{f,j}x_j\leq n_{1,f}+n_{2,f}x_f,\ \forall f\in F-l,\\
&\qquad \sum_{j\in S_l}c_{f,j}x_j\leq n_{1,l},\\
&\qquad x_j\in\{0,1\},\ \forall j\in S_f,\ f\in F,\\
&\qquad x_f\in\{0,1\},\ \forall f\in F-l.
\end{aligned} \tag{3.25}$$

If $\bar{R}_{r,\{l\}}<b_r$, then both b_r and $a_{r,l}$ can be reduced by the slack $b_r-\bar{R}_{r,\{l\}}$.

Problem (3.25) is a SSP with a collection of CECs. In general this problem is likely to be quite difficult to solve, even in special cases, such as when the sets S_f are disjoint, or when $c_{f,j}=1$, for all $j\in S_f$; see [10]. The CECs can be relaxed and the resulting SSP solved, but for large problems it is unlikely that the resulting bound will permit reduction, particularly if many of the knapsack coefficients are small.

Weaker upper bounds can be found by instead formulating easier SSPs, each involving only one set S_f, the knapsack constraint, and the CEC for S_f, as in the

previous section. We let Q_f be the maximum value that the lhs of (1.2) can take if only the variable x_f and those in S_f are allowed to be nonzero, and we let $Q_{1,f}$ be this value if in addition we require $x_f=0$. The values of Q_f and $Q_{1,f}$ are given by the following 0-1 programs.

$$\begin{aligned} Q_f = \text{MAX}\ & a_f x_f + \sum_{j\in S_f} a_j x_j, \\ \text{s.t.}\quad & a_f x_f + \sum_{j\in S_f} a_j x_j \le b, \\ & \sum_{j\in S_f} c_{f,j} x_j \le n_{1,f} + n_{2,f} x_f, \\ & x_j \in \{0,1\},\ \forall j \in S_f,\ x_f \in \{0,1\}; \end{aligned} \tag{3.26}$$

$$\begin{aligned} Q_{1,f} = \text{MAX}\ & \sum_{j\in S_f} a_j x_j, \\ \text{s.t.}\quad & \sum_{j\in S_f} a_j x_j \le b, \\ & \sum_{j\in S_f} c_{f,j} x_j \le n_{1,f}, \\ & x_j \in \{0,1\},\ \forall j \in S_f. \end{aligned} \tag{3.27}$$

Also, we set $Q_{2,f}=Q_f-Q_{1,f}$ and note that this value is always nonnegative. (Note that if $n_{1,f}=0$, then only Q_f is needed; $Q_{1,f}$ will be zero.) Observe that for any 0-1 solution x satisfying (1.2) and (3.24) we have

$$a_f x_f + \sum_{j\in S_f} a_j x_j \le Q_{1,f} + Q_{2,f} x_f, \quad \forall f \in F. \tag{3.28}$$

Thus an upper bound on $R_{r,\{l\}}$ is given by $\bar{R}_{r,\{l\}} = Q_{1,f} + \sum_{f\in F-l} Q_f$. The coefficient reduction can be obtained by using procedure (3.20); that is, for this choice of bounds, all coefficients a_f for $f \in F$ can be reduced simultaneously.

Problem (3.27) is a SSP with an additional knapsack constraint and (3.26) can be put in the same form. We are not aware of specialized algorithms dealing wit this type of problem, but the algorithm given in [3] for solving (or getting upper bounds for) the 0-1 KP with two knapsack constraints can be used.

When all the coefficients $c_{f,j}$ are 0 or 1, the CEC resembles a cover inequality, potentially permitting a more efficient solution; see Section 4. In particular, upper bounds on Q_f and $Q_{1,f}$, say $\bar{Q}_f$ and $\bar{Q}_{1,f}$, respectively, can be obtained by dropping the knapsack constraint and solving the resulting trivial knapsack problems. If $\bar{Q}_f \le b$ then $Q_f = \bar{Q}_f$, and similarly for $Q_{1,f}$. If bounds $\bar{Q}$ are used, then $Q_{2,f}$ in (3.28) must be replaced by the corresponding bound $\bar{Q}_{2,f} = \text{MAX}\{0, \bar{Q}_f - \bar{Q}_{1,f}\}$. Of course, to reduce the coefficient corresponding to l using upper bounds, the bounds must be tight enough so that $\sum_{f\in F} \bar{Q}_f < b + \bar{Q}_{2,l}$. We suggest using upper bounds obtained by considering the two constraints independently, and taking the inferior.

Example 3.4. We conclude this section by presenting some examples. Often con-

sidering only the simple VUB implied by more complex upper bounds will not permit coefficient reduction. For example suppose we have the knapsack constraint (3.29) and the upper bounds (3.30). Here $n_{1,f}=0$ and all the $c_{f,j}$ coefficients are 1.

$$3x_{11}+3x_{21}+5x_{31}+4x_{12}+6x_{22}+3x_{32}+4x_1+6x_2\leq 15. \tag{3.29}$$

$$x_{11}+x_{21}+x_{31}\leq 2x_1, \qquad x_{12}+x_{22}+x_{32}\leq x_2. \tag{3.30}$$

The simple VUBs (3.31) are, of course, implied by (3.30).

$$\begin{aligned} &x_{11}\leq x_1, \qquad x_{21}\leq x_1, \qquad x_{31}\leq x_1,\\ &x_{12}\leq x_2, \qquad x_{22}\leq x_2, \qquad x_{32}\leq x_2. \end{aligned} \tag{3.31}$$

If we consider only (3.29) and (3.31) but not the additional information contained in (3.30), then we have $Q_1=15$ and $Q_2=15$, so the method in Section 3.2 cannot be applied. If instead we use the tighter constraints (3.30), then we have $Q_1=12$ and $Q_2=15$, thus we reduce the coefficient of x_2, producing (3.32). Here $\Delta=15-12=3$.

$$3x_{11}+3x_{21}+5x_{31}+4x_{12}+6x_{22}+3x_{32}+4x_1+3x_2\leq 12. \tag{3.32}$$

Since $Q_1=12$ we cannot reduce further. Observe that the solution (3.33) is feasible for (3.29)–(3.30) but not for (3.32).

$$\begin{aligned} &\bar{x}_{11}=1, \qquad \bar{x}_{21}=\tfrac{1}{2}, \qquad \bar{x}_{31}=0, \qquad \bar{x}_1=\tfrac{3}{4},\\ &\bar{x}_{12}=\bar{x}_{22}=\tfrac{1}{6}, \qquad \bar{x}_{32}=\tfrac{1}{3}, \qquad \bar{x}_2=\tfrac{2}{3}. \end{aligned} \tag{3.33}$$

Example 3.5. As a final example consider the knapsack constraint and bounds given by (3.34)–(3.35).

$$3x_{11}+3x_{21}+5x_{31}+4x_{12}+6x_{22}+x_{32}+4x_1+6x_2\leq 22. \tag{3.34}$$

$$x_{11}+x_{21}+2x_{31}\leq 1+2x_1, \qquad 2x_{12}+x_{22}+3x_{32}\leq 2+3x_2. \tag{3.35}$$

We can reduce the coefficient of x_1 by 3 and the coefficient of x_2 by 4, obtaining the constraint

$$3x_{11}+3x_{21}+5x_{31}+4x_{12}+6x_{22}+x_{32}+x_1+2x_2\leq 15. \tag{3.36}$$

To show that this cut removes a fractional solution, observe that (3.37) gives a solution that is feasible for (3.34) and (3.35) but which violates (3.36).

$$\begin{aligned} &x_{11}=x_{21}=\tfrac{1}{2}, \qquad x_{31}=x_1=\tfrac{3}{4},\\ &x_{21}=\tfrac{1}{2}, \qquad x_{22}=1, \qquad x_{32}=x_2=\tfrac{1}{2}. \end{aligned} \tag{3.37}$$

4. Reducing sets of coefficients

So far we have discussed some methods for computing upper bounds that can be used to reduce an individual coefficient. Now, we will present methods for com-

puting upper bounds for larger sets C. In particular, we consider knapsack constraints of the form (1.2) and cover inequalities of the form (1.5). Let $R_{r,C,i}$ be the maximum value that the lhs of (1.2) can achieve if we require that the cover inequality (1.5) holds with strict inequality and the ith constraint in (1.1) is satisfied. Then, $R_{r,C,i}$ can be obtained by solving the following two constraints 0-1 program

$$\begin{aligned} R_{r,C,i} &= \text{MAX} \sum_{j \in J_r} a_{r,j} x_j, \\ \text{s.t.} \quad & \sum_{j \in J_i} a_{i,j} x_j \leq b_i, \\ & \sum_{j \in C} x_j \leq k_C - 1, \\ & x_j \in \{0,1\}, \ \forall j \in J_r \cup J_i \end{aligned} \tag{4.1}$$

where $C \subseteq J_r$. An upper bound, say $\bar{R}_{r,C}$, on $R_{r,C}$ is given by

$$\bar{R}_{r,C} = \text{MIN}\{R_{r,C,i} \mid i \in I_{r,C}\}. \tag{4.2}$$

We choose

$$I_{r,C} = \{r\} \cup \left\{ i \,\middle|\, M_{i,C} + \sum_{j \in (J_i \cap J_r) - C} a_{i,j} + \sum_{j \in J_i - J_r \mid a_{i,j} < 0} a_{i,j} \geq \eta b_i \right\} \tag{4.3}$$

where η is a specified tolerance and $M_{i,C}$ is the sum of the $k_C - 1$ greatest elements in $\{a_{i,j} \mid j \in C\}$.

Let $\Delta_{r,C}$ be given by (2.6) where $R_{r,C}$ is replaced by $\bar{R}_{r,C}$ (4.2). Let also $k \equiv k_C$. If the original coefficients are integer and $\Delta_{r,C}$ is not a multiple of k, then the reduced coefficient constraint (2.8) will have noninteger coefficients. Taking the integer round-down of the coefficients $\bar{a}_{r,j}$ and appropriately adjusting $\bar{b}_r$ may weaken the constraint. Multiplying through by k will clear fractions without weakening the constraint, but repeated application can lead to arbitrarily large coefficients. If $\Delta_{r,C} > k$, we can replace $\Delta_{r,C}$ by the smaller value $k\lfloor \Delta_{r,C}/k \rfloor \geq 1$.

The reduction procedure can also be applied when the cover inequality is not explicitly included in the original 0-1 program, but is instead implied by other constraints, or even by the constraint (1.2) itself; see [8, 12, 27]. However, for large values of $\Delta_{r,C}$ the coefficient reduction may be so large that (2.8) no longer implies the cover inequality.

A strong reduction can be obtained by exploiting cliques that do not intersect with the set $J_r \cup J_i$. This strategy is called *double coefficient reduction* in our computational report (see below).

Example 4.1. Consider the knapsack constraint

$$10x_1 + 10x_2 + 3x_3 + 3x_4 \leq 15 \tag{4.4}$$

and the implied cover inequality

$$x_1 + x_2 \leq 1. \tag{4.5}$$

Then $R_{\{1,2\}}=6$, so we get $\Delta_{\{1,2\}}=9$ and obtain the constraint

$$x_1+x_2+3x_3+3x_4\leq 6. \tag{4.6}$$

This new constraint does not imply the cover inequality (4.5), and so simply replacing (4.4) by (4.6) results in a weaker nonequivalent 0-1 program.

It is possible to select a smaller, positive value of $\Delta_{r,C}$ so that (2.8) still implies the cover; see [13] for details. However we recommend maintaining a list of detected cover inequalities.

For the example above, choosing $\Delta_{\{1,2\}}=2$, produces the new constraint

$$9x_1+9x_2+3x_3+3x_4\leq 13. \tag{4.7}$$

Finally, by applying twice the individual coefficient reduction procedure (see Section 3.1) on (4.7), we obtain

$$8x_1+8x_2+3x_3+3x_4\leq 11. \tag{4.8}$$

Note that (4.8) is equivalent to and tighter than (4.4), and thus implies the cover inequality (4.5).

Observe that the solution (4.9) is allowed by (4.4) but is infeasible for (4.8).

$$x_1=x_2=0.5, \qquad x_3=x_4=0.7. \tag{4.9}$$

On the other hand, reducing individual coefficients in constraint (4.6) produces

$$x_1+x_2+2x_3+2x_4\leq 4. \tag{4.10}$$

The fractional solution (4.11) is not permitted by constraints (4.4), (4.5) or (4.8), but is feasible for (4.10).

$$x_1=x_2=1, \qquad x_3=x_4=0.5. \tag{4.11}$$

The LP relaxation of (4.1) can be used as a (weak) upper bound on $R_{r,C}$; it has the same complexity as the classical 0-1 LP KP. Let us assume $r=i$. (To simplify notation let us drop the index r.) For $k_C=1$, computing R_C (4.1) consists of solving a SSP with $|J-C|$ variables. For $k_C\geq 2$, if C has sufficiently few subsets having k_C-1 or fewer elements, one may determine $\bar{R}_C$ by solving a collection of similar SSPs (4.1), some of which do not need to be explicitly optimized. For $k_C=2$ we can use the procedure outlined in Section 5.1 to avoid explicit optimizations. In general, we have developed an implicit enumeration like procedure for solving the SSP with a cover inequality. Basically, our procedure consists of implicitly enumerating the nodes defined by the partial solutions associated with the feasible subsets of C, and obtaining the optimal R_C-value of the subproblem attached to each not-yet-fathomed node. A key element in our scheme is the mechanism for detecting that a partial solution will not generate a better solution than the current incumbent; in such a case we do not need to compute the related R_C-value. See [13] for the detailed description of our approach.

As an alternative dynamic programming and branch-and-bound algorithms, or the hybrid algorithm described in [30, 32] can be modified to include a cover constraint. Nodes with more than $k_C - 1$ variables in C fixed at one are immediately fathomed as infeasible. This approach is attractive but we favor our procedure; it uses smaller SSPs, the solution of many of them can be obtained without explicitly solving them and, in any case, the optimal solution of a problem can be used as the initial solution for the next one since the value of the rhs is the unique element to be changed.

In addition, we have found that it is generally more efficient to consider all cover inequalities having large intersection with J_r sequentially, because often one can easily determine the optimal solution of R_{C_1} from the optimal solution of R_{C_2}. Suppose that we have the knapsack constraint (1.2), and two cover inequalities, say

$$\sum_{i \in C_1} x_i \leq k_{C_1} \equiv k_1, \tag{4.12}$$

$$\sum_{i \in C_2} x_i \leq k_{C_1} \equiv k_2. \tag{4.13}$$

Let (SSP_1) and (SSP_2) denote the SSPs with the side constraints: (4.1) with (4.12) and $k_1 - 1$ as its rhs, and (4.1) with (4.13) and $k_2 - 1$ as its rhs, respectively, and let R_1 and R_2 denote the optimal values of these two problems. The problems (SSP_1) and (SSP_2) are quite similar. It is always possible to obtain a feasible solution for (SSP_2) from the optimal solution x^1 of (SSP_1). If $\sum_{j \in C_2} x_j^1 \leq k_2 - 1$ then x^1 itself is feasible for (SSP_2). Otherwise, a feasible solution for (SSP_2) can be obtained from x^1 by setting $\sum_{j \in C_2} x_j^1 - (k_2 - 1)$ of the variables $\{x_j \mid j \in C_2,\ x_j^1 = 1\}$ to zero. This feasible solution gives a lower bound on R_2 and may expedite the exact solution method. In particular, if $R_1 = b$ and x_1 is feasible for (SSP_2) then $R_2 = b$ and there is no need to solve (SSP_2).

Now consider the case where $\Delta_1 = b - R_1 > 0$. We apply the reduction procedure to obtain

$$\sum_{j \in J - C_1} a_j x_j + \sum_{j \in C_1} \left(a_j - \frac{\Delta_1}{k_1}\right) x_j \leq b - \Delta_1. \tag{4.14}$$

Let us assume that x^2 is the optimal solution to (SSP_2). Then, it satisfies (4.14) and, hence, x^2 is feasible, but not necessarily optimal to the following problem

$$\begin{aligned} \bar{R}_2 = \text{MAX} & \sum_{j \in J - C_1} a_j x_j + \sum_{j \in C_1} \left(a_j - \frac{\Delta_1}{k_1}\right) x_j, \\ \text{s.t.} & \sum_{j \in J - C_1} a_j x_j + \sum_{j \in C_1} \left(a_j - \frac{\Delta_1}{k_1}\right) x_j \leq b - \Delta_1, \\ & \sum_{j \in C_2} x_j \leq k_2 - 1, \\ & x \in \{0, 1\}. \end{aligned} \tag{4.15}$$

This is the SSP that is used to further reduce coefficients. (Recall $R_2 = \sum_{j \in J} a_j x_j^2$.)

Note that from (4.15) we have

$$b - \Delta_1 \geq \bar{R}_2 \geq R_2 - \frac{\Delta_1}{k_1} \sum_{i \in C_1} x_i^2. \tag{4.16}$$

Note also that $R_2 < b$ for $\sum_{j \in C_1} x_j^2 \leq k_1 - 1$ since, otherwise, x^2 gives the optimal solution to (SSP_1) and, then, $\Delta_1 = 0$.

In any case, (4.16) can be useful to assess the strength of the reduction based on the inequality (4.13) and so, there is no reason to solve (4.15) as it cannot yield a further reduction, whenever the following condition is satisfied

$$b - \Delta_1 = R_2 - \frac{\Delta_1}{k_1} \sum_{i \in C_1} x_i^2. \tag{4.17}$$

Example 4.2. We first consider an individual knapsack constraint and perform coefficient reductions based on cover inequalities implied by this constraint. Consider the following constraint in 0-1 variables taken from [6].

$$65x_1 + 64x_2 + 41x_3 + 22x_4 + 13x_5 + 12x_6 + 8x_7 + 2x_8 \leq 80. \tag{4.18}$$

Since the coefficients of x_1 and x_2 are very similar, and significantly larger than the other coefficients in the constraint, (4.18) is likely to be equivalent to a constraint having the same coefficients for x_1 and x_2. We attempt to identify such a constraint by considering covers that include x_1 but not x_2. Since $65 + 41 > 80$ we have the cover inequality

$$x_1 + x_3 \leq 1. \tag{4.19}$$

We compute $R_{\{1,3\}} = 79$ (obtained by setting $x_2 = x_5 = x_8 = 1$, $x_6 = x_7 = 0$). Then $\Delta_{\{1,3\}} = 1$, so we reduce the rhs of (4.18) and the coefficients of x_1 and x_3 each by 1, obtaining the constraint

$$64x_1 + 64x_2 + 40x_3 + 22x_4 + 13x_5 + 12x_6 + 8x_7 + 2x_8 \leq 79. \tag{4.20}$$

Note that this constraint has an odd rhs and only one odd coefficient, so the variable x_5 must take the value one in every solution satisfying (4.20) at equality. We apply the coefficient reduction procedure with the trivial cover inequality

$$x_5 \leq 1. \tag{4.21}$$

We compute $R_{\{5\}} = 78$, $\Delta_{\{5\}} = 1$, and obtain the constraint

$$64x_1 + 64x_2 + 40x_3 + 22x_4 + 12x_5 + 12x_6 + 8x_7 + 2x_8 \leq 78. \tag{4.22}$$

In (4.22), all the coefficients and the rhs are even, so we divide all coefficients by 2 to obtain

$$32x_1 + 32x_2 + 20x_3 + 11x_4 + 6x_5 + 6x_6 + 4x_7 + 1x_8 \leq 39. \tag{4.23}$$

Note that $32 + 32 > 39$ and consider the implications of the cover inequality

$$x_1 + x_2 \leq 1. \tag{4.24}$$

Since $R_{\{1,2\}}=38$, and $\Delta_{\{1,2\}}=1$, we obtain the equivalent constraint

$$31x_1+31x_2+20x_3+11x_4+6x_5+6x_6+4x_7+1x_8\leq 38. \tag{4.25}$$

The clique $J=\{1,2,3\}$ gives $R_{\{1,2,3\}}=28$, $\Delta_{\{1,2,3\}}=10$ and results in the constraint

$$21x_1+21x_2+10x_3+11x_4+6x_5+6x_6+4x_7+1x_8\leq 28. \tag{4.26}$$

For the clique $J=\{1,2,4\}$ we compute $R_{\{1,2,4\}}=27$, so $\Delta_{\{1,2,4\}}=1$ and we obtain the constraint

$$20x_1+20x_2+10x_3+10x_4+6x_5+6x_6+4x_7+1x_8\leq 27. \tag{4.27}$$

Once again we have an odd rhs and only one odd coefficient, so we use the trivial constraint $x_8\leq 1$ to obtain $R_{\{8\}}=26$ and $\Delta_{\{8\}}=1$, producing the tighter equivalent constraint

$$20x_1+20x_2+10x_3+10x_4+6x_5+6x_6+4x_7\leq 26. \tag{4.28}$$

We again divide through by 2 to obtain

$$10x_1+10x_2+5x_3+5x_4+3x_5+3x_6+2x_7\leq 13. \tag{4.29}$$

The coefficient reduction procedure in [6] produces the equivalent constraint

$$4x_1+4x_2+2x_3+2x_4+1x_5+1x_6+1x_7\leq 5. \tag{4.30}$$

Note that the fractional solution (4.31) is infeasible for (4.29), but is permitted by (4.30) and the original constraint (4.18). On the other hand (4.32) is infeasible for (4.18), (4.30) and the constraint induced by the clique $\{1,2\}$, but is permitted by (4.29).

$$x_1=0.75,\qquad x_2=x_3=x_4=0,\qquad x_5=x_6=1,\qquad x_7=x_8=0. \tag{4.31}$$

$$x_1=1,\qquad x_2=0.3,\qquad x_3=x_4=x_5=x_6=x_7=x_8=0. \tag{4.32}$$

The system given by (4.29) and the constraints induced by the cliques $\{1,2,3\}$ and $\{1,2,4\}$ is equivalent to and tighter than the constraints (4.18) and (4.30).

Example 4.3. The following 0-1 program has been used in [36] to show the effectiveness of LP based cutting planes procedures for solving 0-1 programs. It is shown that appending the most violated cover induced inequalities to each (enlarged) LP relaxation of the 0-1 original problem (4.33), only four LP problems are required to obtain the optimal solution. (No branch-and-bound mechanism was needed.)

$$\begin{aligned}
&z=\text{MAX } 77x_1+6x_2+3x_3+6x_4+33x_5+13x_6+110x_7+21x_8+47x_9,\\
&\text{s.t.}\quad 774x_1+76x_2+22x_3+42x_4+21x_5\\
&\qquad +760x_6+818x_7+62x_8+785x_9\leq 1500,\\
&\qquad 67x_1+27x_2+794x_3+53x_4+234x_5\\
&\qquad +32x_6+797x_7+97x_8+435x_9\leq 1500,\\
&\qquad x_j\in\{0,1\},\ j=1,2,\ldots,9.
\end{aligned} \tag{4.33}$$

The optimal solution of the LP relaxation is given by (4.34), and (4.35) gives the optimal solution of the original program (4.33).

$$z_{\mathrm{LP}} = 225.7 \quad \text{with } x_1 = 0.71,\ x_2 = 0,\ x_3 = 0.35,\ x_4 = x_5 = 1,\ x_6 = 0,$$
$$x_7 = x_8 = 1,\ x_9 = 0. \tag{4.34}$$

$$z = 176 \quad \text{with } x_1 = 0,\ x_2 = 1,\ x_3 = 0,\ x_4 = x_5 = 1,\ x_6 = 0,\ x_7 = x_8 = 1,$$
$$x_9 = 0. \tag{4.35}$$

By using our procedure for identifying maximal cliques [15] and solving (4.1), we obtain the following set of clique inequalities that can replace the set of knapsack constraints in (4.33).

$$x_1 + x_6 + x_7 + x_9 \leq 1, \qquad x_3 + x_7 \leq 1. \tag{4.36}$$

Note that in this case we do not need to solve any LP problem. By inspection we obtain the optimal solution (4.35).

5. Increasing coefficients

5.1. Motivation

Coefficient increasing and/or constraint lifting as outlined in Section 2, see (2.12)–(2.19), can be performed by using a methodology very similar to the coefficient reduction mechanism. We have found that one of the most useful strategies is to increase cover or knapsack coefficients by sequentially using a subset of knapsack constraints in addition to a (possibly, trivial) clique. The auxiliary problems are also (hopefully, small) KPs and SSPs as in Section 4.

For the case where C is a (possibly, trivial) clique the resulting auxiliary problem is a pure 0-1 KP. In this case, the KPs for obtaining $S^p_{r,C,i}$ and $S^q_{r,C,j}$ for $p, q \in C$ are very similar. They differ only in one element, x_p is fixed to 1 in the first problem, say (KP_p), and x_q is fixed to one in the second problem, say (KP_q). The optimal solution to (KP_p) is a feasible solution to (KP_q), whenever $a_{r,p} > a_{r,q}$. On the other hand, $S^q_{r,C,r} = S^p_{r,C,r} + \delta$, where $\delta = a_{r,q} - a_{r,p}$, whenever $0 \leq \delta \leq b_r - S^p_{r,C,r}$, and so, the SSP for obtaining $S^q_{r,C,r}$ does not require the explicit solution.

5.2. Tightening cover inequalities

In this section we demonstrate the coefficient increase procedure by applying it to cover inequalities. These covers can be explicitly included in (1.1) or implied by (1.1), in which case the lifted cover rather than the original cover will be appended. In either case we denote the cover constraint by r, the coefficients by $c_j \in \{0, 1\}$ and the rhs by $b_r < \sum_{j \in J} c_j$. That is, we increase coefficients of the inequality

$$\sum_{j \in J} c_j x_j \leq b_r. \tag{5.1}$$

We compute upper bounds on $S^l_{r,C}$ by considering knapsack constraints from (1.1). That is, we solve the 0-1 program

$$\begin{aligned} S^l_{r,C,i} = \text{MAX} & \sum_{j \in G} c_j x_j, \\ \text{s.t.} & \sum_{j \in G} a_{i,j} x_j \leq b_i, \\ & x_l = 1, \\ & x_j = 0, \ \forall j \in C - \{l\}, \\ & x_j \in \{0,1\}, \ \forall j \in G \end{aligned} \tag{5.2}$$

where $G \equiv J_r \cup J_i$, $a_{i,j} = 0$ for $j \in J_r - J_i$, C is a (possibly trivial) clique such that $C \subset G$, and let $l \in C$. (Note: The clique inequality in (5.2) is redundant whenever it is implied by the ith knapsack constraint.) An upper bound $\bar{S}^l_{r,C}$ on $S^l_{r,C}$ is given by

$$\bar{S}^l_{r,C} = \text{MIN}\{S^l_{r,C,i}, \ i \in I_{r,C}\}, \tag{5.3}$$

where we choose the set of constraints $I_{r,C}$ as in (4.3).

For each clique C we may generate a constraint (2.19) by using the procedure (2.17)–(2.18), where $S^l_{r,C}$ is replaced by $\bar{S}^l_{r,C}$; any one of these new constraints can replace the original.

A strong increase can be obtained by exploiting cliques that do not intersect with the set G. This strategy is called *double coefficient increase*.

Our procedure for computing these upper bounds on S is very efficient, primarily because the 0-1 program (5.2) is a knapsack problem with all objective function coefficients equal to 1. For a given constraint i, each bound $S^l_{r,C,i}$ can be computed in $O(\log J_i)$, once the coefficients $a_{i,j}$ have been sorted; see [16].

Example 5.1. Consider the following constraint in 0-1 variables taken from [37].

$$103x_1 + 71x_2 + 66x_3 + 66x_4 + 66x_5 + 65x_6 + 38x_7 \leq 205. \tag{5.4}$$

By using our procedure described in [15] we identify the following cover inequality implied by (5.4).

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \leq 3. \tag{5.5}$$

The constraint (5.5) is obtained as an extension of the minimal cover $\{4, 5, 6, 7\}$. By applying the procedure outlined above (see [16] for the details), to (5.4) and (5.5), we obtain the constraint

$$2x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \leq 3. \tag{5.6}$$

Note that $103 + 38 < 205 < 103 + 38 + 65$. In this case the new constraint (5.6) is facet defining; see [37].

Example 5.2. Consider again the following constraint in 0-1 variables taken from [6].

$$65x_1 + 64x_2 + 41x_3 + 22x_4 + 13x_5 + 12x_6 + 8x_7 + 2x_8 \leq 80. \tag{5.7}$$

By using our procedures described in [14, 15] we identify the system given by (5.8) and the cliques $\{1,2,3\}$ and $\{1,2,4\}$. The new system is equivalent to and tighter than (5.7); see Example 4.2. We also identify the cover inequality (5.9).

$$10x_1 + 10x_2 + 5x_3 + 5x_4 + 3x_5 + 3x_6 + 2x_7 \leq 13. \tag{5.8}$$

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 3. \tag{5.9}$$

By applying our procedure to the knapsack constraint (5.8), the cover inequality (5.9) and the clique $\{1,2,3\}$, we obtain the new constraint

$$2x_1 + 2x_2 + x_3 + x_4 + x_5 + x_6 \leq 3. \tag{5.10}$$

Note that the fractional solution (5.11) is feasible for the original constraint (5.7) as well as for its replacement (constraint (5.8) and the inequalities induced by the cliques $\{1,2,3\}$ and $\{1,2,4\}$). It is also feasible for the cover inequality (5.9). On the other hand, it is infeasible for the tightened cover inequality (5.10).

$$x_1 = x_2 = 0.35, \qquad x_5 = x_6 = 1, \qquad x_3 = x_4 = x_7 = 0. \tag{5.11}$$

Example 5.3. Consider also the following constraint in 0-1 variables taken from [34].

$$24x_1 + 22x_2 + 20x_3 + 18x_4 + 18x_5 + 68x_6 + 34x_7 + 16x_8 + 16x_9 + 12x_{10} + 25x_{11} \leq 85. \tag{5.12}$$

By using a procedure described in [15] we identify the cliques $\{(i,6)\}$ for $i = 1,2,3,4,5,7,11$ and the covers

$$(8,2,1,11,7,6), \tag{5.13}$$

$$(9,2,1,11,7,6), \tag{5.14}$$

$$(4,2,1,11,7,6), \tag{5.15}$$

$$(5,2,1,11,7,6), \tag{5.16}$$

$$(3,2,1,11,7,6), \tag{5.17}$$

all with rhs $b_r = 3$ (5.1).

By using the tightening procedure we can get $\bar{a}_6 = 2$ in (5.13) and (5.14), and $\bar{a}_6 = 3$ in (5.15), (5.16) and (5.17).

We also identify the cover

$$(8,9,4,5,3,2,1,11,7,6) \tag{5.18}$$

with $b_r = 4$. The procedure cannot tighten (5.18), but it drops x_8 and by using the resulting subcover it obtains $\bar{a}_6 = 3$ and $\bar{a}_7 = 2$. The new valid constraint is given by

$$x_1+x_2+x_3+x_4+x_5+3x_6+2x_7+x_9+x_{11}\leq 4. \tag{5.19}$$

Similarly, by dropping both x_8 and x_9 the procedure obtains $\bar{a}_6=4$ and $\bar{a}_7=2$. The resulting constraint (5.20) is facet defining, as is (5.19); see [34].

$$x_1+x_2+x_3+x_4+x_5+4x_6+2x_7+x_{11}\leq 4. \tag{5.20}$$

If we drop the variable indexed by $\{9,8,4,5,3\}$, we can increase the coefficient of x_{11} to 2.

We also identify the covers

$$(j,5,3,2,1,11,7,6) \quad \text{for } j=10,8,9,4 \tag{5.21}$$

with $b_r=4$, but any constraint generated by the procedure is dominated by (5.20).

6. Computational experience

An extensive computational experience is underway to assess the effectiveness of the new procedures. Here we report some experience with an implementation of some of our reformulation procedures. Our methodology has been embedded in a cutting plane based scheme for tightening the LP relaxation of the 0-1 program (see [8, 24] among others), and afterwards, a branch-and-bound (henceforth, b&b) strategy is used.

We report results for two sets of problems. Set $S1$ consists of random cases to test the performance of our approach on nonstructured problems. Set $S2$ consists of four real-life cases, two of them are benchmark problems taken from [8], see also [27].

The main steps of our current implementation are as follows.

(1) Fix variables based on trivial feasibility considerations.

(2) Reverse negative coefficients to improve reformulating performance. (Note: Pairs of 0-1 variables and equality constraints may be introduced.)

(3) Identify maximal cliques and subsets of (lifted) minimal covers, and create a candidate cover list. See [15].

(4) Tighten cover inequalities as in Section 5.2.

(5) Replace knapsack constraints by tighter equivalent constraints obtained by performing coefficient reduction (by exploiting cliques and covers) as in Section 4.

(6) Obtain a bound on the optimal solution of the 0-1 program by solving the LP relaxation of the current model. We use IBM OSLv1.1 [26].

(7) Select cliques and (possibly, tightened) covers from the candidate list that are violated by the current LP solution, append the induced inequalities to the current model, and go to step (6) if any constraint has been appended.

(8) Perform double coefficient reduction on the knapsack constraints that are satisfied as equalities by the current LP solution, and go to step (6) if any constraints are tightened.

Table 1: LP computational experience. Problem's set $S1$

Case	Size			Active		Obj fun value			Value reduct %		
	m	n	*den* %	*NR*	*NC*	$z(L)$	$z(K)$	$z(I)$	*LGAP*	*KGAP*	*RK*
P1	20	50	5	7	19	17.0	15.7	15	11.7	4.5	65.0
P2	20	50	50	3	20	303	289	289	4.6	0.0	100
P3	20	50	14	11	10	20.7	20.4	18	13.2	11.7	0.88
P4	20	100	9	6	30	1905	1850	1816	4.7	1.8	62.1
P5	50	50	30	2	24	349	314	277	20.6	11.8	51.4
P6	50	50	50	10	11	10.4	9.9	8	23.1	20.0	79.1
P7	50	100	10	9	55	47	44.2	43	8.5	2.8	70.2
P8	100	100	30	10	40	534	515	458	40.6	11.1	75.0
P9	250	100	10	3	73	737	615	471	36.2	21.3	46.2

(9) Delete redundant cuts. The optimal solution of the (possibly, enlarged) model is obtained by using the standard b&b strategy of OSLv1.1. (Note: Currently, no cuts are appended to the subproblems attached to b&b nodes. Plans have been announced for including in OSLv1.2 a powerful preprocessing phase as well as a device for adding global and local cuts at selected nodes.)

Tables 1 and 2 report some results for set $S1$. They give information about the new cuts, integrality gap, LP objective function deterioration, and b&b performance. The tables' columns are as follows: m, # of original constraints; n, # of variables; *den*, constraints matrix density; *NR*, # of active tightened knapsack constraints; *NC*, # of active new cuts; $z(L)$ and $z(K)$, optimal solution values of the LP relaxation of the original model and new model, respectively; $z(I)$, optimal solution value of the 0-1 program; *LGAP*, integrality gap of the LP relaxation of the original model, defined as $|z(L)-z(I)|/z(L)$; *KGAP*, integrality gap of the LP relaxation of the new model, defined as $|z(K)-z(I)|z(K)$; *RK*, gap reduction due to our model's

Table 2: # of branch-and-bound nodes. Problem's set $S1$

Case	Opt sol for model		Optimality proof		
	Original	New	*NN*	*NNR*	*NR*
P1	423	201	3191	201	93.7
P2	13	0	19	0	100.0
P3	291	362	5828	3078	47.2
P4	1264	358	9774	466	95.2
P5	520	62	3308	242	92.7
P6	65	466	>10000	621	-
P7	4147	20	7391	277	96.2
P8	6880	1391	>10000	1438	-
P9	-	79	>10000	4918	-

Table 3: LP computational experience. Problem's set $S2$

Case	Size			Active		Obj fun value			Value reduct %		
	m	*n*	*den* %	*NR*	*NC*	*z(L)*	*z(K)*	*z(I)*	*LGAP*	*KGAP*	*RK*
*C*1	69	180	4.1	23	6	1267	1212	1200	5.3	1.0	82.1
*C*2	184	515	2.3	86	4	2093	2069	2069	1.1	0.0	100
*C*3	177	548	1.8	47	158	315	7104	8691	265	22.3	81.0
*C*3a	177	548	1.8	35	435	315	8374	8691	265	3.8	96.2
*C*4	147	2655	3.4	12	11	6532	6535	6548	0.24	0.21	12.5
*C*4a	147	2655	3.4	22	43	6532	6535	6548	0.24	0.20	18.8

reformulation, defined as $|z(L)-z(K)|/|z(L)-z(I)|$; *NN* and *NNR*, number of b&b nodes that are required for proving optimality for the original model and new model, respectively; *RN*, % reduction in # of nodes due to our model's reformulation, defined as $(NN-NNR)/NN$.

No reduced cost fixing mechanism was used for set $S1$. A limit of 10000 nodes has been imposed. The impact of model reformulation in reducing the number of nodes is clearly demonstrated.

The columns of Tables 3 and 4 have the same meaning as in Tables 1 and 2, respectively. Problems $C1$ and $C2$ are production scheduling problems. Problems $C3$ and $C4$ are production planning benchmark problems; some computational results are reported in [8], see also [27].

The integrality gap of $C1$ was drastically reduced (82.1%) by using our reformulation. This permitted the b&b strategy to prove optimality in 1178 nodes. This same strategy was not able to find any integer solution for the original model in the allowed number of nodes.

Our reformulation approach gets the optimal value $z(I)$ of $C2$ without b&b, but it still needed 161 nodes to find the integer solution. On the other hand, the integrality gap of the original model was only 1.1%, but the b&b strategy had not found any integer solution in 4000 nodes (at which point the search was interrupted).

We used two different solution strategies for the benchmark problems. The rows

Table 4: # of branch-and-bound nodes. Problem's set $S2$

Case	Opt sol for model		Optimality proof		
	Original	New	*NN*	*NNR*	*NR*
*C*1	-	1063	>10000	1178	-
*C*2	-	161	>4000	161	-
*C*3	-	95	>10000	480	-
*C*3a	467	32	1031	76	92.6
*C*4	80	14	264	48	81.8
*C*4a	43	21	161	28	82.6

labeled $C3$ and $C4$ correspond to the strategy described above, namely, b&b on the original model is compared to applying our reformulation procedures and then using b&b on the new model. Our model reformulation achieves a 81% reduction on the integrality gap of $C3$; the optimality was proved at b&b node #480. The integrality gap of the original model of $C3$ is well known, so it was not a surprise that b&b could not obtain any solution to the original model in the allowed number of nodes. (We are aware of a parallel experiment conducted by the OSL team, where the optimal solution of $C3$ is achieved without b&b, by using a carefully designed preprocessing phase for variable fixing based on feasibility and optimality considerations and generating violated cliques, covers and VUB induced inequalities among others.)

Our reformulation achieves a 12.5% reduction on the integrality gap of $C4$, and on the reformulated model the b&b strategy needs only 18.2% of the number of nodes required for the original model.

Our second scheme for the benchmark problems replaces OSL by MPSX [25]. MPSX includes a preprocessing phase (see also [8]) with the following main steps:

(1) Variable fixing due to feasibility and optimality considerations.

(2) Solving the LP relaxation of the current 0-1 model.

(3) Identify and append the following constraints, provided that they are violated by the current LP solution: constraints obtained by performing coefficient reduction on knapsacks by using trivial cliques (see [27] and Section 3.1), VUB implied by aggregate constraints (see [8, 22]), and cliques and (lifted) minimal covers implied by knapsack constraints.

(4) Reduced cost fixing.

The rows labeled $C3$a and $C4$a contain the results for problems $C3$ and $C4$, respectively, obtained using our reformulation approach and MPSX preprocessing in an interactive mode until no more LP solution value deterioration is achieved. As in the study in [8], we used 0.10 as the threshold for reduced cost fixing.

Applying only the MPSX preprocessing to the original model of $C3$ and then using the b&b strategy used in [8], optimality was proven at node #1031. By using our reformulation approach together with MPSX preprocessing before beginning the same b&b strategy, only 76 nodes were needed to prove optimality. The approach described in [8], which uses the preprocessing methods now included in MPSX as well as a procedure for generating violated $(1, k)$-configuration induced inequalities (not yet included in MPSX), required only 36 nodes to prove optimality for this problem.

The methods in [8] required 214 nodes to prove optimality for problem $C4$. Surprisingly, when MPSX preprocessing alone was applied, the optimality was proved in 161 nodes. Our reformulation together with the MPSX preprocessing phase requires only 28 nodes to prove optimality. We are not aware of any other experiment that requires smaller number of nodes. The synergetic effect of combining a state-of-the-art preprocessing phase with our reformulation approach is apparent from this experiment.

Conclusions

New methods for efficient reformulation of 0-1 programs have been described. The basic idea consists of taking advantage of the special structure of constraints that either are given in the model or can be implied by the same constraints system. The additional information that we are currently using for reformulating 0-1 knapsack constraints and cover inequalities is provided by the trivial 0-1 constraints on the variables, variable upper bounding constraints, capacity expansion constraints, clique and cover inequalities and other 0-1 knapsack constraints, these procedures can be used to produce cuts when the conditions required by myopic methods are not satisfied. The reformulation consists of either reducing the variables coefficients and the rhs of the constraint, or increasing the variables coefficients. Although in some instances our methods require solving some 0-1 programs (mostly, 0-1 knapsack problems and subset-sum problems), the models are usually small; they involve only a small subset of the original variables. We have shown easy ways to find upper bounds for R and S. There are likely to be many other easy ways, particularly when special structures are available. The resulting cuts are not as deep as those generated with optimal values, but can eliminate fractional solutions when myopic methods cannot be applied.

We have shown, by theoretical proofs as well as by experimenting with examples taken from the open literature, how the new procedures may generate tighter equivalent formulations. We have reported computational experience on several cases, two of them well-known real-life problems; the results are quite promising. An extensive computational experimentation is underway to assess the effectiveness of the new procedures.

Acknowledgement

We wish to thank Monique Guignard, Ellis Johnson, Silvano Martello, Kurt Spielberg, Paolo Toth and Laurence Wolsey. Their constructive suggestions have significantly improved the paper.

References

[1] E. Balas, Facets of the knapsack polytope, Math. Programming 8 (1975) 146–164.

[2] E. Balas, On determining whether a valid inequality is facet defining, in: R. Kannan and W.R. Pulleyblank, eds., Integer Programming and Combinatorial Optimization (Waterloo University Press, Waterloo, Ont., 1990) 45–59.

[3] J. Barcelo and E. Fernandez, Problemas de knapsack 0-1 con una restriccion addicional, Qüestiió 12 (1988) 175–208.

[4] E.M.L. Beale and J.A. Tomlin, Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables, in: J. Lawrence, ed., Operations Research 69 (Tavistock, London, 1970) 447–454.

[5] E.A. Boyd, Polyhedral results for the precedence-constrained knapsack problem, in: R. Kannan and W.R. Pulleyblank, eds., Integer Programming and Combinatorial Optimization (Waterloo University Press, Waterloo, Ont., 1990) 85–100.
[6] G.H. Bradley, P.L. Hammer and L. Wolsey, Coefficient reduction for inequalities in 0-1 variables, Math. Programming 7 (1974) 263–282.
[7] V. Chvatal, Edmonds polytopes and a hierarchy of combinatorial problems, Discrete Math. 4 (1973) 305–357.
[8] H. Crowder, E.L. Johnson and M.W. Padberg, Solving large-scale zero-one linear programming problems, Oper. Res. 31 (1983) 803–834.
[9] B.L. Dietrich and L.F. Escudero, Coefficient reduction for knapsack constraints in 0-1 programs with VUBs, Oper. Res. Lett. 9 (1990) 9–14.
[10] B.L. Dietrich and L.F. Escudero, More coefficient reduction for knapsack constraints in 0-1 programs with VUBs, RC 14389, IBM Research, T.J. Watson Research Center, Yorktown Heights, NY (1989).
[11] B.L. Dietrich and L.F. Escudero, New procedures for preprocessing 0-1 models with knapsack like constraints and conjunctive and/or disjunctive VUBs, RC 14572, IBM Research, T.J. Watson Research Center, Yorktown Heights, NY (1989).
[12] B.L. Dietrich and L.F. Escudero, Obtaining clique, cover and coefficient reduction inequalities as Chvatal–Gomory inequalities and Gomory fractional cuts, RC 14707, IBM Research, T.J. Watson Research Center, Yorktown Heights, NY (1989).
[13] B.L. Dietrich and L.F. Escudero, Coefficient reduction with cover inequalities, RC 14913, IBM Research, T.J. Watson Research Center, Yorktown Heights, NY (1989).
[14] B.L. Dietrich and L.F. Escudero, On extended coefficient reduction for 0-1 programs, RC 15013, IBM Research, T.J. Watson Research Center, Yorktown Heights, NY (1989).
[15] B.L. Dietrich and L.F. Escudero, On identifying maximal cliques and extensions of minimal covers implied by 0-1 knapsack like constraints, RC 15960, IBM Research, T.J. Watson Research Center, Yorktown Heights, NY (1990).
[16] B.L. Dietrich and L.F. Escudero, On tightening cover induced inequalities, European J. Oper. Res., to appear.
[17] B.L. Dietrich and L.F. Escudero, On strengthening the formulation of the workload allocation problem for parallel unrelated machines with setups, Internat. J. Flexible Manufacturing Systems, to appear.
[18] L.F. Escudero, S3 sets. An extension of the Beale–Tomlin special ordered sets, Math. Programming 42 (1988) 113–123.
[19] D. Fayard and G. Plateau, An algorithm for the solution of the 0-1 knapsack problem, Computing 28 (1982) 269–287.
[20] D. Freville and G. Plateau, Heuristics and reduction methods for multiple constraints 0-1 linear programming problems, European J. Oper. Res. 24 (1986) 206–215.
[21] R.E. Gomory, Outline an algorithm for integer solutions to linear programs, Bull. Amer. Math. Soc. 64 (1958) 275–278.
[22] M. Guignard and K. Spielberg, Logical reduction methods in zero-one programming (minimal preferred variables), Oper. Res. 29 (1981) 49–74.
[23] P.L. Hammer, E.L. Johnson and U.N. Peled, Facets of regular 0-1 polytopes, Math. Programming 8 (1975) 179–206.
[24] K. Hoffman and M.W. Padberg, LP-based combinatorial problem solving, Ann. Oper. Res. 4 (1985) 145–194.
[25] IBM, MPSX, Mathematical Programming System Extended/370, Program Reference Manual, SH19-6553 (1988).
[26] IBM, OSL, Optimization Subroutine Library, Guide and Reference, SC23-0519 (1990).
[27] E.L. Johnson, M.M. Kostreva and U.H. Suhl, Solving 0-1 integer programming problems arising from large-scale planning models, Oper. Res. 35 (1985) 803–819.

[28] F. Kianfar, Tighter inequalities for 0,1 integer programming using knapsack functions, Oper. Res. 19 (1971) 1374–1392.
[29] F. Kianfar, Tighter inequalities for 0-1 integer programming: Computational refinements, Oper. Res. 24 (1976) 581–585.
[30] S. Martello and P. Toth, A mixture of dynamic programming and branch-and-bound for the subset-sum problem, Management Sci. 30 (1984) 765–771.
[31] S. Martello and P. Toth, A new algorithm for the 0-1 knapsack problem, Management Sci. 34 (1988) 633–644.
[32] S. Martello and P. Toth, Knapsack Problems: Algorithms and Computer Implementations (Wiley, New York, 1990).
[33] R.K. Martin and L. Schrage, Subset coefficient reduction cuts for 0/1 mixed-integer programming, Oper. Res. 33 (1985) 505–526.
[34] G.L. Nemhauser, G. Sigismondi and P. Vance, A formula for finding facet-defining lifted cover inequalities, Report #J-89-06, School of Industrial and Systems Engineering, Georgia Institute of Technology Atlanta, GA (1989).
[35] G.L. Nemhauser and L.A. Wolsey, Integer and Combinatorial Optimization (Wiley, New York, 1988).
[36] G.L. Nemhauser and L.A. Wolsey, Integer programming, in: G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd, eds., Handbooks in Operations Research and Management Science, Vol. I: Optimization (North-Holland, Amsterdam, 1989) 447–527.
[37] M.W. Padberg, A note on zero-one programming, Technical paper, New York University (1973); see also: Oper. Res. 23 (1975) 833–837.
[38] M.W. Padberg and G. Rinaldi, A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, num. 319, Laboratoire d'Econometrie de l'Ecole Polytechnique, Paris (1989).
[39] U.H. Suhl, Solving large-scale mixed-integer programs with fixed charge variables, Math. Programming 32 (1985) 165–182.
[40] J.A. Tomlin, Special ordered sets and an application to gas supply operation planning, Math. Programming 42 (1988) 69–84.
[41] T. Van Roy and L.A. Wolsey, Solving mixed integer programming problems using automatic reformulation, Oper. Res. 35 (1987) 45–57.
[42] H.P. Williams, Model Building in Mathematical Programming (Wiley, New York, 1978).
[43] J.M. Wilson, Generating cuts in integer programming with families of special ordered sets, European J. Oper. Res. 46 (1990) 101–108.
[44] L.A. Wolsey, Facets for a linear inequality in 0-1 variables, Math. Programming 8 (1975) 165–178.
[45] L.A. Wolsey, Tight formulations for mixed integer programming: A survey, Math. Programming 45 (1989) 173–191.
[46] E. Zemel, Easily computable facets of the knapsack polytope, Math. Oper. Res. 14 (1989) 760–764.